嵌入式操作系统应用丛书

μC/OS ARM 移植要点详解

黄燕平 编著

北京航空航天大学出版社
http://www.buaapress.com.cn

嵌入式操作系统应用丛书

μC/OS ARM移植要点详解

黄燕平　编著

北京航空航天大学出版社
http://www.buaapress.com.cn

内 容 简 介

本书适合的读者是对 ARM 微处理器有一定了解，对嵌入式内核有一定了解和对嵌入式产品开发有一定经验的读者。对于从事嵌入式产品开发，特别是基于 ARM 的嵌入式产品开发的项目经理、体系结构设计师、设计师、代码开发工程师、测试工程师，解决实际问题有一定的帮助。

本书内容内容共 7 章，各章主题如下：

第 0 章为嵌入式环境的选择，对嵌入式产品开发中常见的芯片、软件方案进行了简单比较分析。

第 1 章为 OS 内核概念，包括 ARM 微处理器特性、内核结构基础等重要概念的详细说明。它是本书中非常重要的一章。

第 2 章为 μC/OS - II 移植过程，是在常见 ARM 微处理器上移植 μC/OS - II 的代码详解。

第 3 章为代码组织及功能设计，把嵌入式产品的设计从简单移植的角度扩展到内核整体体系结构设计及功能组件组织的角度并引入一个有益的、重要的 COS 组件方法。它是本书中篇幅最长的一章，也是最重要的一章。

第 4 章为 μRtos V1.0 代码说明，介绍一种硬实时分层调度体系结构的嵌入式内核产品。

第 5 章为 ARM 开发环境，解答软件开发工具使用中的一些常见问题。

第 6 章为软件工程简述，对嵌入式产品开发中的软件项目管理中的要点进行了探讨，讨论了一些如何提高产品品质的技术知识。

图书在版编目(CIP)数据

μC/OS ARM 移植要点详解/黄燕平编著. —北京：北京航空航天大学出版社，2005.11

ISBN 7 - 81077 - 725 - 4

Ⅰ. μ… Ⅱ. 黄… Ⅲ. 微处理器，ARM—系统设计

中国版本图书馆 CIP 数据核字(2005)第 113709 号

μC/OS ARM 移植要点详解

黄燕平 编著

责任编辑 王 鸿

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083) 发行部电话：010－82317024 传真：010－82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

涿州市新华印刷有限公司印装 各地书店经销

*

开本：787×1 092 1/16 印张：17.25 字数：442 千字

2005 年 11 月第 1 版 2005 年 11 月第 1 次印刷 印数：5 000 册

ISBN 7 - 81077 - 725 - 4 定价：26.00 元

前 言

随着国内工业化、数字化的步伐加快，嵌入式开发在IT行业中的重要性越来越显著。

中国成为“世界制造中心”甚至“设计中心”的趋势，必然导致对小型数字控制系统的需求越来越大。在嵌入式系统开发方面，最核心的技术就是微处理器芯片和嵌入式操作系统。其中在微处理器芯片方面，ARM已经给出了比较理想的一个答案；而在嵌入式操作系统方面，适合国内发展方向的解决方案以及系统基础结构方面并不理想。

- 风河公司的VxWorks操作系统成本高，结构复杂，不适合小型应用。
- 微软公司的WinCE操作系统更适合民用、便携式娱乐设备等。
- 开源的Linux操作系统体系结构同样复杂，产品化和商业化程度不够，即使在Linux本来的PC目标环境下，也难寻理想的技术支持，更不用说嵌入式环境下的Linux。这方面的弱势对批量生产、大规模、长时间运行使用的工业化产品来说是致命的。

另外，在以上讨论的这3种系统中，只有VxWorks是硬实时操作系统，而WinCE和Linux是非硬实时操作系统。

在这种情况下，类似于μC/OS－II的小型硬实时嵌入式操作系统内核具有低成本、易控制、小规模、高性能的特性，因而有相当好的发展前景。但是这类系统的基础较为薄弱，面临产品化和商业化程度不够的局面。采用此类系统进行产品开发需要仔细分析和设计，否则也很难真正满足工业产品生产的要求。

本书正是针对这种情况，在ARM微处理器环境下，针对商业化、产品化环境的严格要求，设计、构造了一种硬实时嵌入式内核体系结构。当然，真正的商业化、产品化的嵌入式内核既需要这种能够满足高标准要求的体系结构设计基础，又需要严格的产品化软件开发测试过程。只有理论基础与工程实践完整地结合，才能产生真正经受得起考验的，能够满足工业化生产，能够在各种环境下稳定运行并确保达到设计目的的产品。从这个角度考虑，仅仅拿来一个操作系统内核并开发应用产品很难完全满足这种要求。必须要对内核的设计思路进行仔细的考虑和验证，对应用的可选开发设计方法进行审慎的评估，并配合真正工业化的项目开发管理办法，才能保障产品达到要求。

本书中提到的μRtos V1.0内核，正是作者及其所在团队按照以上精神付出巨大努力严格设计、测试的产品。该内核的体系结构设计思路在本书中有充分详细的解释和说明。另外，μC/OS－II是读者在市面上可以方便获得的一种“半开

源”的操作系统内核。本书针对该内核在ARM下的移植以及与本书所述内核体系结构的关系及比较，进行了详细解说。通过对比，既方便μC/OS-II的爱好者、使用者学习掌握μC/OS-II内核，同时又在对比分析过程中，使读者掌握μC/OS-II和μRtos V1.0内核各自的详细特征、特点，方便读者在此基础上开发设计出更好的嵌入式系统产品。

作　者

2005年7月25日

致　谢

轮回

——谨将此诗献给 μRtos 内核及各位读者

这一路不容易，
记忆穿行过地狱，
当火烧灼灵魂的时候，
我用火去采集露滴。

这一路不容易，
灭入寂静等待着孕育，
守着内心露水中的须弥，
命运是否从此谱续？

本书在编写过程中得到了上海的程民军先生等的大力支持，在此特别感谢帮助作者完成本书的各位好友、出版编辑等。

程民军先生与作者在深圳共事多年，与作者的讨论经常激发作者的灵感，他对本书中各章节进行了细致审阅，并提出了宝贵的意见。

同时，还要感谢参与本书内容讨论的多位网友，作者并不清楚他们中大多数人的名字，能记得的只是一些网络代号。虽然没有在此将他们的网名一一列出，但是对他们的感谢同样是由衷的。

也许本书内容还很难达到各位朋友理想中的目标，但是其中凝聚了各位朋友、同事的心血和智慧，他们的鼓励推动着作者将本书完成。也许在将来的某个时候，作者能够给这些朋友、同事一个更为满意的答案。

目　录

第0章 嵌入式环境的选择

0.1 简 介

本书针对的读者是已经有一些嵌入式开发经验的工程师。读者应该已经了解 ARM 内部寄存器的基本用法，已经对 μC/OS－II 有一些基本了解。掌握这两项基本知识的最好途径是阅读 μC/OS－II 原作者 JEAN J. LABROSSE 所著的《嵌入式实时操作系统 μC/OS－II 第 2 版》(邵贝贝译)[1]以及杜春雷编著的《ARM 体系结构与编程》[2]。

主要内容包括 3 个方面：一是构造嵌入式操作系统内核；二是在 ARM 环境下移植和改进 μC/OS－II 内核(包括网络协议栈移植等)；三是介绍 μRtos V1.0 操作系统。本书以第一方面的内容为主线，其他两方面的内容作为第一方面内容的具体实例进行解说。

读者最关心的问题可能是 μC/OS－II 系统移植的重点、难点在什么地方？有些读者第一印象是汇编代码部分的移植。实际从事嵌入式系统开发/移植工作的读者应该能体会到，汇编代码部分的移植虽然有一些难度，但从完整产品的角度来看，这部分工作难度并不大。如果仅仅是尝试 JEAN J. LABROSSE 在其书中列举的几个简单例子，或其他类似的学习体验式的开发，甚至可以说非常容易。只有在面对商业化产品的要求，考虑更广泛或更长远的问题时，才会面对更多的问题。例如：硬实时特性是否有保障？中断响应能力如何？面对各种设备驱动有明确的开发接口规范保障驱动能和整个体系完整地融合吗？系统级的服务应用(例如网络服务)应该遵循一个什么样的体系？上层应用应该遵循一个什么样的体系？

嵌入式开发领域从早先的以 8 位单片机设计开发为主的时代开始进入 32 位微处理器占据大部分市场的时代，各方面的环境都在发生变化，市场的需求也在发生变化。当前，嵌入式开发领域对产品的要求不仅仅体现在功能要求越来越多，而且其他各方面的要求也越来越高。可以明显感觉到的包括如下几个方面。

通信速率：嵌入式产品对通信速率的要求大幅度提高，以前低速的串行通信逐渐被高速串行通信替代，并且开始转变为以高速的网络通信手段为主。

稳定性：嵌入式产品对运行稳定性的要求大幅度提高，以前小范围专用系统所需要的稳定性要求被广泛采用的系统稳定性要求替代。更大范围的使用对产品的稳定性要求是有本质变化的。

产品功能：嵌入式产品对实用功能的要求大幅度提高，以前嵌入式产品的功能大多单一、简单，现在综合性的产品功能要求随处可见。例如，经常见到的数字产品已具备了图形化界面，加上了大容量存储系统等，甚至还要能够具有很好的语音、图像处理识别等智能型功能。

这些环境的变化，无不对嵌入式产品的开发提出了更高的要求。这种要求是多方面的，包括对硬件的要求，也包括对软件的要求，还有对产品开发的工程组织方面的要求。

这个行业并不缺少能满足这方面要求的产品，但主要涉及到成本、代价等多方面问题。例如，著名的 VxWorks 嵌入式实时操作系统就是人们经常提到的可用产品之一。VxWorks 主要针对的是各种高端应用，例如经常被举例的火星行走者项目和各种高端路由器产品等。VxWorks 面市时，与高端应用对应的民用产品的开发环境主要还处于 8 位单片机的时代，32 位微处理器的民用化才刚刚起步。而现在随着 ARM 及类似等级的芯片在民用产品中的快速普及，在民用、普通工业、低成本控制系统等方面，开发人员的首选并不是 VxWorks 这种体系复杂的产品。其他一些解决方案，如 WinCE、linux 改造版的方案也都存在这样的问题。这些产品都来自于一个强大的背景，但是在竞争激烈中，低成本、灵活性、性能等各方面的要求驱动着开发设计人员寻找更适合他们的产品方案。

ARM 和 μC/OS - II 的组合有希望成为这样一种方案，但是这种方案能否承载起这么大的希望，还需要澄清或解决多方面的问题。这需要从更严格、更基础的角度来审视这样一种选择。也就是必须认真地从商业化产品开发的角度来进行一些更基础的工作。

在 ARM/μC/OS - II 的组合中，ARM 微处理器方面不是本书关注的要点，本书主要关注操作系统内核方面的问题。μC/OS - II 的特点是从不多的几个实时内核概念出发，发展出一套简洁、可用的算法，在很小的内核代码中完成了抢占式任务调度、任务间通信等高级功能。正因为代码规模极小，涉及到的概念清晰、简单，开发人员更容易掌握和控制。使用此类系统开发产品，工程师在实际产品开发过程中也更容易根据自己的需要灵活地进行调整，制作出自己的高性能、低成本的产品。这里的低成本并不是指产品中采用廉价芯片这样的概念，读者应该理解当前 IT 行业产品开发的主要成本是在开发的人力资源投入上。对于极小规模的内核，当然投入的人力要少很多，后期的维护成本也要低很多。即便产品出现问题，也更容易定位解决。毕竟，一个一万行左右代码规模的内核和一个十万行代码规模的内核，在定位、解决问题时，工作量经常不止 10 倍的关系。何况 μC/OS - II 的内核代码还不到一万行的规模。如果这么小规模的内核基础牢固，完全值得信赖，接口简单好用，那么一定是产品开发工程师的福音。

μC/OS - II 的目的是要提供一个嵌入式开发的基础平台。从后来提供的 μcGUI、μcFS、μcIP、μcFLASH 这样一个系列的发展中也可以明显体会到 μC/OS - II 这种基础的地位。正是这种基础的地位，更需要仔细的审核。如果基础不牢，无疑是沙上建房，事倍功半。

通过商业化产品开发的过程，本书作者体会到 μC/OS - II 主要提供的几组内核算法存在一些问题，有的是移植作者的问题，有的是内核具体代码的问题，有的是体系结构方面比较严重一点的问题。其中最显著的两点是：一是硬实时特性没有充分保障；二是没有提供一个完整的设备驱动框架。

本书同样尝试从几个简单、基本的实时内核设计概念出发，以 μC/OS - II 类似的代码规模，提供一种适合在 32 位微处理器环境下使用的内核。这样做的目的并非仅仅针对 μC/OS - II的个别不足之处，作者希望能为本行业同事提供一个经得起仔细测试审核的基础牢固，接口简单好用，值得信赖的基本平台。同时，在与 μC/OS - II 的对比中，为 μC/OS - II 的用户提供移植改善的经验，以便读者能够将自己的 μC/OS - II 用得更好。由于作者的能力所限，本书中难免出现不足，甚至错误之处，敬请读者指正。

虽然 μC/OS - II 中简单谈到编写设备驱动的方法，但是离一个合用的完整体系结构还有相当的距离。JEAN J. LABROSSE 针对嵌入式系统常见设备驱动另外编写了一本著作

Embedded Systems Building Blocks[3]（《嵌入式系统构件》），但还是没有为 μC/OS－II 构造一个完整的设备驱动体系。这也是使用 μC/OS－II 进行开发时能感觉到的与其他嵌入式内核相比差距最大的地方。

例如，在 Linux 中编写驱动只须写不多的几个固定的函数，并且每个函数的功能都有仔细的定义（参见 Alessandro Rubini 的《Linux 设备驱动程序》[4]）。VxWorks 也有类似的框架体系；WinCE 内核中驱动的体系更为复杂，但同样是完整的。

如果是编写非常简单的设备驱动或者上层应用，这个问题并不突出，当复杂性提升或希望不同产品线保持体系结构上的一致性时，这个问题就比较显著。特别是当前嵌入式系统的复杂性越来越高的情况下（例如添加网络协议栈已经非常普及），这方面的问题明显影响了 μC/OS－II 的推广和应用。规划一个简单、稳定、规范的体系结构在当前已经非常必要。

这样一个体系结构并不需要复杂的定义，应该具有高效、简洁、易扩展、易裁剪等特点，应该在关键的几个方向上进行有预见性的定义。一旦制定好这样一个体系，μC/OS－II 或类似的其他嵌入式操作系统的高效和简洁的特点才能够极大地发挥出来。

针对这种现状，本书并不仅仅解说如何编写高效的代码，更多地是从更基本的起点开始，构造一个抢占式多任务、单一内存空间、非微内核特点的体系结构框架，并且保持类似 μC/OS－II 的简洁高效特征。μC/OS－II 和 μRtos V1.0 是这种特征内核两个具体实例。μC/OS－II 是 JEAN J. LABROSSE 的作品，μRtos V1.0 是作者开发的一个嵌入式操作系统产品。

在市场上可获得的嵌入式操作系统内核商业产品中，VxWorks V5.4 是此类嵌入式操作系统的典型代表，但是 VxWorks 比较复杂、庞大，代价也极高，比较适合高端的产品应用（例如 VxWorks 应用在美国航天局的火星行走器上）。VxWorks 后来的版本已经发生了变化，针对高端应用采用了内存空间保护方式，已经不再是本书讨论的这个方向上的产品。不过，作者打算在不远的将来，配合 ARM9 芯片在 μRtos V2.0 中增加简单的内存地址空间保护特性。

抢占式多任务、单内存空间、非微内核的嵌入式操作系统，具有高效、紧凑等特点，特别适合小型专用嵌入式产品场合使用。其他方案的嵌入式操作系统中通常会考虑微内核、多内存空间保护等，适应的应用环境不同。例如，对于有大量第三方程序存在的环境，内存空间保护是一个不错的选择，微内核方案则特别适合于比较复杂的系统同时在不同微处理器间移植要求又比较突出的情况。

本书所选择的方向有能力扩展到包括其他方案的特征或特点。例如，这种方案也可以在一定程度上扩充多内存空间保护的特点，还可以通过移植 POSIX 接口或 TAO，让第三方程序的移植和开发更简便。具体如何实现这些特点，将在本书的后续章节中讨论。实际上，不同的嵌入式内核方案，彼此都有借鉴、扩充的倾向。例如，μCLinux 在 ARM7 环境下，为了提高效率（当然也是为了适应 ARM7 的环境），去掉了内存空间保护部分，rtLinux 为了提高 Linux 系统的实时性，在内核中又包含一个硬实时内核，借用了微内核的一些手法，但与通常微内核体系并不相同。

嵌入式开发涉及到比较多的硬件知识，本书中除了软件方面的内容外，同时包括一些与软件相关的微处理器等硬件方面的特性。对于起辅助作用的硬件方面的内容，本书中进行了简略介绍，原则上是希望有一定基础的读者能够比较流畅地阅读，不必过多纠缠于不必要的环节。而在技术要点方面，本书尽量详细描述，对原理、具体实现等相关方面进行仔细分析。不过“详解”类型的书籍难免在一些细节上会比较繁琐，可能没有仅仅解释概念然后列出代码那

种编写方式简洁、清爽。本书比较强调从“设计”阶段的实现角度仔细阐述各种内核概念的实际设计实现手法，因此很多章节的内容看起来像产品中的设计文档的解说，有很多的接口定义和伪代码，而本书内容也正是主要来自于作者所开发商业产品的设计资料。部分章节为了具体举例提供了实际代码，更多的代码读者可以从作者提供的网站上获得。除 μC/OS－II 移植部分的内容和一些需要举例详细说明的内容外，很多设计方案都只给出了接口定义，读者可以和网站上获得的代码仔细对照。作者始终觉得在书中除了解说举例需要的代码外，没有必要将内容变成代码解说教材。少部分的概要设计资料结合主要的详细设计资料和少部分的代码解说，应该是与读者沟通的最佳方式。

本书中的基本概念类型的内容集中在第 1 章，这一章是为构造内核、移植改进 μC/OS－II 准备概念和原理知识的。阅读该章对于高级读者虽然不是必需的，但是作者还是推荐读者首先从第 1 章开始阅读。第 1 章中提出了一些后续实现的思路和体系结构方案，并不仅仅是单纯的概念介绍。特别是 1.1.2 小节及以后的全部内容，请读者仔细关注。

商业开发不同于普通试验和学习式开发，其特点是设计开发过程更为严谨，并且需要完整的设计文档记录和测试记录。多数爱好者在对 μC/OS－II 进行研究式移植、开发时并没有进行认真的设计工作，包括没有对 μC/OS－II 进行分析设计的工作。从事学习式开发的一些读者仅仅是在读过 μC/OS－II 原著后，参照其中的移植指导就开始移植。有些读者仅仅是下载到对应自己所使用微处理器的 μC/OS－II 移植版本后，用调试工具跟踪并把其中的影响系统基本运行的 BUG 去除后，就开始写自己的测试应用。这种开发移植方式缺乏对 μC/OS－II 的充分认识，缺乏对其优点、缺点的了解。想要在此基础上构造稳定高效的商业化产品，无疑是很不现实的。希望通过分析“嵌入式操作系统内核构造”，详细描述嵌入式内核方面的基础资料，同时扩充读者对 μC/OS－II 特性的了解。

商业开发不同于普通学习式开发的另一个特点是严格的测试，特别是作为基础的内核，其牢固程度必须有明确的验证手段的证明。嵌入式系统相对 PC 系统中的应用来说，通常面对的运行环境要苛刻很多。即便只是移植使用 μC/OS－II 这种的系统，在没有严格测试的情况下作为商业化产品使用也是不严谨的。产品要达到高标准的要求，离不开测试方案。各层次测试方案都是在分析、设计文档资料基础上产生的，因此，即便是拿来主义的移植使用，如果不制作、不准备详细的分析并设计文档，就很难准备出好的测试方案并进行测试。本书靠后的章节，给出了软件测试方案设计的技术要点详解，并结合本书前面章节的分析资料，给出了一些制作嵌入式操作系统测试方案的指南，方便读者在了解 μC/OS－II、μRtos V1.0 的同时了解软件测试的技术，并在认为必要时，自己通过实际的测试验证不同系统各方面的特性。

本书选择当前具有代表性的环境作为讨论问题的基本平台，结合实际产品进行技术要点的详细解说。平台方案选择的微处理器是三星公司的 S3C44B0 和飞利浦公司的 LPC2214，内核采用 μC/OS－II 或 μRtos V1.0，网络协议栈采用 LWIP。下面简单说明与这个平台相关的几个方面。

0.2 关于微处理器

选择三星 S3C44B0 主要是因为其廉价、易得，读者自己的产品开发完全可以选用其他 ARM 芯片，并不影响本书的适应性。目前，S3C44B0 在参考板、学习环境等场合使用较多。

选择飞利浦 LPC2214 的主要原因同样是因为其廉价、易得。飞利浦公司的 ARM 处理器在工业、商业产品中使用较多。

ARM 是当前嵌入式领域使用最广泛的微处理器。ARM 芯片种类很多，都是采用英国 ARM 公司(前身是 Acorn 公司)设计并授权的 RISC 内核，主要的 ARM 制造厂商包括 INTEL、ATMEL、飞利浦、三星等。ARM 内核的微处理器从 ARM7 到最新的 ARM11 等，具有高性能、低价格、通用性好等特点。最常见的是 ARM7 和 ARM9 这两个系列，ARM7 通常用于低成本方案，ARM9 通常用于性能要求较高的方案。各厂家根据自己的目标不同，在 ARM 内核的基础上扩充了不同的协处理器、外围接口等，方便嵌入式产品的设计。ARM 系列的微处理器已经完全不同于 8 位单片机时代的微处理器，性能提高很大，价格并不高，而软件系统的通用性相对单片机环境要好很多，开发模式也有极大改善。

ARM7 是 3 级流水线 RISC 内核，以 MIPS(百万指令每秒)表示的 ARM7 运算能力指标的计算公式是：主频×0.9。也就是说平均一条指令的执行只需一个时钟周期略多一些的时间，约 20 ns(在 50 MHz 主频条件下)。5 级流水线 RISC 内核的 ARM9 的 MIPS 指标计算公式是：主频×1.1。也就是说，一条指令的执行平均不到一个时钟周期就可以完成。50 MHz 主频的 ARM7 的 MIPS 指标约为 45 MIPS(参见具体处理器的数据手册)，该指标已经超过 i486 的性能指标。这也是 μCLinux 等从 PC 环境的操作系统衍生来的系统能够在 ARM7 这类微处理器上流畅运行的原因。通常这些 ARM 芯片内还包含了网络等极为丰富的外设，而价格却非常低。而主频能达到 200～400 MHz 的 ARM9 系列微处理器，例如 S3C2410 等，性能已经超过 i586(Pentium)，外设更加丰富，性价比更好。

本书中涉及到性能参数方面的数据，除非特别说明，均是按照 50 MHz 的 ARM7 微处理器环境给出的。如果采用的微处理器不同或者主频不同，读者可以自行推算。要提醒读者注意的是，如果产品配置的是低速内存，则性能会有很大差别。

各种 ARM 微处理器都带有丰富的特殊功能寄存器，用于控制系统和操作外部设备等功能。外设是否丰富，特殊功能寄存器设计是否合理、好用是选择微处理器的重要标准。从特殊功能寄存器设计的合理性角度来考察，飞利浦公司的各系列 ARM 微处理器更值得推荐，但是很遗憾的是飞利浦当前缺少带网络接口的 ARM 微处理器系列。对照三星和飞利浦系列微处理器的详细使用手册，读者就能够对两种处理器设计思路的不同有清晰的认识。

本书作者采用 SDT V2.51 进行开发，交叉调试接口是 JTAG。同时配合一些小工具，还具备代码直接烧写到 Flash 的能力。选择这种方案也是考虑为读者提供一个最低成本的开发平台。如果采用仿真器，当然功能更强大，但成本要高很多，而且必要性并不显著。这样一个开发环境包括了代码编写、编译、链接、调试和烧写的全部功能，简单、好用。

0.3　关于 OS

μC/OS－II 内核作为一种代码公开的(并非通常意义的开源)嵌入式实时操作系统内核非常有特色，在规模不大的代码内实现了抢占式任务调度、多任务间通信等功能，任务调度算法也很独特。该内核裁剪到最小状态后编译出来只有 8K 左右，全部内核功能(添加 LWIP 网络协议栈等)也就 100K 左右，资源消耗非常小。市面上一些 ARM 微处理器片上所带内存就已经足够一个裁剪合适的内核的简单应用，非常方便产品的开发设计。

纯商业性的嵌入式操作系统主要有 WinCE 和 VxWorks，开源的嵌入式操作系统有 μCLinux，半开源性质的嵌入式操作系统有 rtLinux、μC/OS－II、μRtos V1.0 等。WinCE 比较适合实时性要求不高但通用性要求较高的产品（如消费型 PDA），VxWorks 比较适合于复杂性、重要性要求都比较高的大型商业化应用（例如核心路由器），Linux 系列的衍生产品（μCLinux 和 rtLinux 等）适合比较复杂但重要性程度不高的低成本产品，μRtos V1.0 适合实时性要求较高、成本控制要求也较高的商业化系统（如设备控制、通信、家电等）。从实时控制这个角度来说，μRtos V1.0 的性能表现要比 μCLinux、WinCE 等操作系统更好。

当前，μC/OS－II 是一个基本完整的嵌入式操作系统解决方案套件，包括 μC/TCP－IP（IP 网络协议栈）、μC/FS（文件系统）、μC/GUI（图形界面）、μC/USB（USB 驱动），μC/FL（Flash 加载器）等部件。但是这些部件不是公开代码的，因此本书不准备深入讨论。μRtos V1.0 也同样具有协议栈、文件系统、GUI、常用端口驱动等套件，并且具备更为完整的体系结构规范。

还有一些比较重要的可能在嵌入式环境中发挥重要作用的部件，包括嵌入式数据库、POSIX 兼容性接口、常用设备的驱动模块等。将来这个行业里还会产生更多的重要部件需求，在互联网上的开源社区通常能够找到相应的开源代码包，并可以进行移植。一般为开源社区或嵌入式应用准备的代码包在开发过程中都很好地考虑了移植接口问题。简单移植通常没有太大的问题，条件是首先熟悉操作系统内核的接口及内部机制，并熟练运用。但是严谨的商业产品设计开发中的移植工作还是要花费大量的研究工作的，特别是要准备分析资料和测试方案。

μC/OS－II 原有代码支持 64 个任务，通过简单的改造就能够支持 1 024 个任务。μRtos V1.0 原则上没有限制任务数，与系统可供使用的内存资源相关，还与采用的内核算法相关，因为内核算法是可以替换的。Linux 操作系统早期同样只支持 1 024 个进程（那个时期的 Linux 还不支持线程），并且是在 i386 环境下承担着运行多种用户程序的责任。现在，常见 ARM7 系列芯片通常都能达到或接近 i586 芯片的性能，因此当前条件下在 μC/OS－II、μRtos V1.0 环境运行多种复杂功能部件是完全可行的。

小型嵌入式产品的解决方案以前主要采用的是 8/16 位单片机。用单片机开发产品，除非是非常小的应用；否则实际产品生产成本可能比 ARM 微处理器还高，而且性能差距很大。因此，当前嵌入式开发环境已经发生了很大变化，已经没有太多必要在嵌入式系统中保留对 8/16 位单片机环境的兼容性，专门针对 32 位微处理器构造嵌入式系统可以更充分地发挥 μC/OS－II、μRtos V1.0 系统的潜力，系统代码也更为简洁、完整。

0.4 关于功能模块的移植

嵌入式环境开发中经常需要面对的一个工作就是功能模块代码的移植，这些功能模块包括网络、文件系统、GUI 等，互联网上的开源社区是这些代码的主要来源，特别是以 BSD、Linux 为代表的开源代码群体。

LWIP 网络协议栈是开源社区的贡献之一，具有高性能、适合嵌入式环境等特点。LWIP 的主要代码衍生自具有悠久历史的 BSD 操作系统中的协议栈。在本书配套的代码中采用这个协议栈，既是应用的需要，也是一个移植大型软件包到 μC/OS－II 和 μRtos V1.0 的范例，

以便读者移植文件系统、图形界面、数据库服务等软件包时参考，同时借此体会、检验体系结构规范所带来的益处。

0.5　关于本书

目录是本书内容的一个索引，读者可以通过目录查找自己想要查看的章节。为了方便读者阅读和检索资料，编制了关于本书内容的主目录和索引本书中所列代码及伪代码的代码/伪代码目录(见附录 B)。

本书中一些内容引用到 μC/OS－II 原作者 JEAN J. LABROSSE 的 *MicroC/OS－II The Real-Time Kernel Second Edition*。该书的中译版名字是《嵌入式实时操作系统 μC/OS－II (第 2 版)》，由邵贝贝等翻译。本书中引用该书资料的内容采用“μC/OS－II 原书…”这样的方式进行说明，并详细注明出处、引用到的页码、章节等。这些引用页码、章节以中译本中的页码、章节为准。μC/OS－II 原书是一本很好的参考资料，详细说明了该操作系统内核的基本特点、代码实现逻辑、基本的使用、移植方法等。本书省略了一些基础性资料，如果读者对这些基本知识感兴趣，推荐读者去查看这本 μC/OS－II 原书。该书所配光盘还包括一个完整的 μC/OS－II V2.52 代码，非常方便读者学习。在网络上也能找到各种 μC/OS－II 的版本，当然其正式网站 http://www.uC/OS－ii.com 上的版本最为丰富。如果仅仅是要找 ARM 方面的移植资料，那么其他网站上的版本也很有特色，并且可能下载到比 μC/OS－II 原书所配代码版本更高的源代码。不过仔细研究这些代码就会发现，V2.52 之后的修订已经非常少，因此 V2.52 是一个不错的学习起点。如果读者手中有通过正式商业渠道获得的 V2.76 或更高版本当然更好。

当提到 μC/OS－II 内核时，经常简称为“μC/OS－II”。当本书中涉及到汇编代码时，采用的是 SDT V2.51 下的 ARM 汇编的语法格式，与 μC/OS－II 原书中 X86 汇编指令有相当大的区别。如果读者不熟悉 ARM 汇编，则可以查找一下其他资料。本书中汇编代码举例的地方并不多，大部分代码是 C 代码。汇编代码格式不同于 C 代码，没有括号包围函数体，注释符号也不同。

读者可以在 http://www.uRtos.net 上下载 μRtos V1.0 公开代码版的完整代码，用于学习、参考。除本书中引用说明他人或其他组织机构拥有版权的资料外，本书作者拥有本书中全部代码、资料，以及这些代码、资料所体现的设计方案等全部可权益化资料的版权。读者或其他人士、组织如果要使用本书作者拥有版权的作品，请与本书作者联系。对未经本书作者授权在商业产品或活动中使用本书作者拥有版权作品的情况，本书作者保留相关法律条款赋予的全部权力。

本书中列出的 μC/OS－II 代码是作者移植、改进后的代码，只要读者购买或下载到 μC/OS－II代码，仔细阅读本书内容后，应该完全能够根据自己的产品需要动手构造出更完整、更高效的 μC/OS－II 系统。

本书中提出了一种称为 COS(Componet Of Source code)的技术，以便在嵌入式环境中规范化代码的移植和组织。IT 业最丰富的代码资源主要都是针对历史悠久的主机计算环境，特别是后来的 PC 机计算环境。嵌入式开发工程师都理解这种状况，并且也都在身体力行地做着从主机计算环境代码向嵌入式计算环境代码的移植工作，主要的代码来源是各种开源代码。

但是这种移植工作当前的状况极为混乱，缺少一种有效的代码组织管理技术。而主机计算环境，特别是 PC 机计算环境中，有类似的技术可以借鉴。例如 COM 和 CORBA 都是主机/PC 计算环境中行之有效的手段。也有嵌入式开发工程师或机构组织在尝试将 COM、CORBA 引入到嵌入式开发环境中。但是直接引入 COM 和 CORBA 对于嵌入式环境来说，资源消耗过高，并且这两种技术都是二进制模块的组织管理技术，并非针对源代码层次的管理。

源代码层次的组织管理虽然可以借助 C/C++语言自身的逻辑方式进行管理，但是这种管理规模过小，通常一个功能模块内部就会有多个 C/C++语言文件。在嵌入式开发环境中，针对源代码的、组件模块级别的管理技术，对嵌入式开发中的源代码管理是最适用，帮助也是最大的。如果这种技术同时能够为代码作者提供产品保护，开发创新代码的工程师能够保护自己的关键成果，则更加理想。

COS 正是本书作者针对以上需求制定的一种技术规范。COS 具有针对源代码组织管理、资源消耗极小、适合模块级代码、可以保护开发人员工作等特性。详细内容请读者参考第 3 章，特别是 3.1 节的内容。

本书除引用 JEAN J. LABROSSE 著作的部分内容外，还参考了其他一些书籍。书中涉及到相关内容时都详细说明了相关书籍的书名、作者等，既是为了帮助读者查找、参考相关书籍，同时也是为了表达作者对这些前辈的尊敬。这些前辈的经验积累对作者的产品开发提供了宝贵的帮助，开拓了作者的视野，同时也激发出作者将产品开发中积累的经验编著成书与诸位同行共享的意愿。希望本书能够达到对读者有益、开拓读者视野、激发读者灵感的目的。